TR-814
DAAG-53-76C-0138

October 1979

TREE AND PYRAMID STRUCTURES FOR CODING HEXAGONALLY SAMPLED BINARY IMAGES.

Peter J. Burt
Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

Technical rept.

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND
20742

(14) TR-814
DAAG-53-76C-0138

(11) October 1979

(6) TREE AND PYRAMID STRUCTURES FOR CODING HEXAGONALLY SAMPLED BINARY IMAGES. (12) 28

(10) Peter J. Burt
Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

(9) Technical rept.,

ABSTRACT

Tree and pyramid type data structures are described which may be used for storing and processing binary images that have been sampled on a hexagonal grid. These are analogous to the quadtrees and pyramids which have recently been developed for images sampled on a rectangular grid. Trees may be formed with 3, 4, 7, 9 ... branches per node. Of these the "septtree", formed with 7 branches per node, promises to be particularly interesting since each node "covers" a roughly hexagonal region of the image.

(15) DAAG53-76-C-0138
DARPA Order-3206

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

1. Introduction

Array data structures are commonly used for storing and processing binary images, with one array cell dedicated to storing the image color (black or white) at each pixel point. Tree data structures may also be used for storing two-dimensional images, and those can be considerably more efficient than arrays, both in computer memory requirements and processing steps, when images contain large areas of a single color.

Tree structures with four branches per node, quadtrees, have been developed for images sampled on a rectangular grid (see, for example, [1,2]). A tree structure with seven branches per node has been developed for images sampled on a hexagonal grid [7]. In this paper we examine a variety of branching patterns for hexagonally sampled images, and show that trees may be defined with 3, 4, 7, 9, 11, 12, and 13 or more branches per node. Of these, 3, 4, 7, and 13 permit patterns that are compact and symmetric under 60 or 120 degree rotations.

The triangular pattern obtained with three branches per node may have an advantages over quadtrees for some applications since image reduction from level to level of the tree is somewhat less, namely 3:1 rather than 4:1. The pattern obtained with 7 branches is hexagonal, so has the advantage that each node has six edge neighbors but no corner neighbors. This uniform neighbor feature is useful for image processing based on local patterns [3,4]. Patterns obtained with a larger number of branches are of less interest since image reduction is very rapid from level to level of the tree.

We consider only those trees in which the branching pattern is the same at each level, and which "cover" the plane, without overlap. The branching pattern of such a tree may be used as a generator pattern for a hierarchy or "pyramid" of hexagonal arrays. Such structures are analogous to pyramids which have been developed for rectangular arrays [5,6]. Generator patterns tessellate the hexagonal grid, and the centroids of the patterns form the nodes of a new hexagonal grid. The pyramid is generated by recursively tessellating the grids formed by previous tessellations.

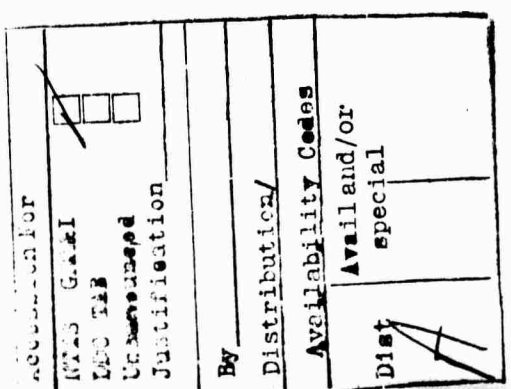
A node may be designated by its pyramid address (i,j,k), where k is the number, or level, of the array in the pyramid and i,j are its coordinates within that array. Alternatively, it may be designated by its tree address $(b_N, b_{N-1},\ldots,b_{k+1})$, which is the sequence of branches one must take to move from the root of the tree, at level N, to the node itself, at level k. The correspondence between equivalent tree and pyramid addresses is not trivial, as it is for the quadtree. Therefore, a procedure will be given for translating one type of address into the other in the case of a seven-branch, hexagonal septtree.

2. Tessellating the Hexagonal Grid

It is well known that hexagonal tiles, which are identical in size and orientation, tessellate a plane surface; they fully cover the surface, without overlap. We shall be concerned here not with tiles on continuous surfaces, but with "patterns" of nodes, on an hexagonal grid. Such a grid is shown in Figure 1a, with a diamond shaped pattern indicated as a group of four nodes connected by lines. A pattern "tessellates the grid" if identical copies of the pattern can be arranged so that they cover all grid nodes, and no node is covered by more than one pattern. The diamond pattern of Figure 1a has this property, as is demonstrated in Figure 1b. Two other tessellations of the grid are shown in Figures 1c and 1d.

The centroids of patterns which tessellate the hexagonal grid form the nodes of a new grid. These are indicated by open circles in Figure 1. If the grid formed by the centroids is itself hexagonal, then it also may be tessellated by the same pattern, and the tessellation process can be repeated recursively. This is the fundamental operation in the formation of a tree or pyramid.

The first objective of the present paper is to discover those patterns which may be used in tree construction. Two constraints on such patterns are already clear:



C1) The pattern should tessellate the hexagonal grid.

C2) The centroids of patterns which tessellate the hexagonal grid should themselves form a hexagonal grid.

Two other constraints may be added to eliminate trees with undesirable properties:

C3) The pattern should have the same orientation everywhere in the tessellation.

C4) The pattern should be "compact". In particular it should only connect nodes which are immediate neighbors in the hexagonal grid.

Note that none of the patterns in Figure 1 satisfy these constraints. While all tessellate the grid (C1), the centroids of the patterns in Figures 1b and 1c do not form a hexagonal grid (C2), the pattern in Figure 1c does not have the same orientation everywhere (C3), and the pattern of Figure 1d is not compact (C4).

3. Derivation of Admissible Patterns

Suppose that the nodes of a hexagonal grid are separated by a unit distance. Consider patterns which connect n nodes. If such a pattern is to satisfy constraint C2, then the distance between nodes in the grid of centroids must be $\sqrt{n}$. This is the case because the new grid will have 1/n as many nodes as the original, yet these nodes must be distributed over the same area.

Next, suppose we label the n nodes in the pattern by letters a, b, c, ... as has been done for the diamond pattern in Figure 1a. Since, by constraint C3, all patterns must have the same orientation, the 'a' nodes in neighboring patterns must also be separated by $\sqrt{n}$.

Let an 'a' node of a pattern coincide with the node numbered '0' in Figure 2a. The 'a' node of a neighboring pattern should then coincide with one of the nodes 1, 2, 3,.... By computing the distance to these nodes we may determine permissible values of n. (We need only consider nodes in a 30° arc centered on node '0' since nodes outside the arc are congruent to nodes inside the arc under appropriate rotations and reflections.) The distances to the first 8 numbered nodes and the corresponding values of n are shown in Table 1.

The data in the table show that pattern sizes 3, 4, and 7 are admissible on the hexagonal grid, while 2, 5, and 6 are not.

Pattern size 1 is admissible but not of interest since the grid of centroids is identical to the original. Patterns of certain larger sizes, 9, 12, 13, 16 ... , are also admissible, but these are rather large for use in trees, and will not be considered further.

The above argument also allows us to determine the exact position of the 'a' nodes in neighboring patterns of any admissible tessellation. These patterns are shown for n = 3, 4, and 7 in Figure 2. The patterns are unique except for translations, rotations or reflections.

Note that the grids formed by the patterns for n = 3 and 7 are tilted with respect to the original grid. The degree of tilt is indicated in Table 1. This tilt is independent of other details of the pattern, which as yet have not been specified.

With the positions of the 'a' nodes known for each n, all of the admissible patterns satisfying constraints C1 to C4 are readily generated. This is done by "growing" a pattern around its 'a' node: nodes 'b', 'c', ... are added one at a time to each instance of the pattern so that each node added to a given pattern must adjoin one neighbor already in the pattern (C4) and not fall on top of any node already in that pattern or in neighboring patterns (C1). Some of these admissible patterns are shown in Figure 3.

Most admissible patterns are rather irregular in shape. These are not well suited for tree construction when we want

high level nodes, or subtrees (these are defined below) to represent local regions of an image. Therefore we may wish to consider only that pattern for each size n which is most compact, i.e. that pattern which has maximum contact among its members and, hence, minimum contact with nodes outside the pattern. These patterns are unique for n = 3, 4, and 7, and are shown in Figure 4a, b, and c. Figure 4d shows an alternative size 4 pattern which is not the most compact pattern of this size, but which has a 3 point "star" symmetry since it is congruent to itself under 120° rotations.

It should be noted that the diamond pattern in Figure 4b is identical to that in Figure 1b, but that the arrangement of neighboring patterns on the grid is slightly different in the two cases. For this reason the centroids of the diamonds in Figure 4 form a hexagonal grid but those in Figure 1 do not.

4. Patterns as Pyramid Generators

The admissible patterns described in the previous section all have the property that when the patterns are used to tessellate the hexagonal grid, their centroids form a new hexagonal grid. Let the original grid be called array 0, and let the new array formed by the centroids of a pattern used to tessellate array 0 be called array 1. Since array 1 is also hexagonal it may be tessellated with the same pattern to form array 2. The procedure may be repeated to form progressively higher order arrays. The set of such arrays form a "pyramid", so called because each higher array has fewer nodes than its predecessor by a factor of 1/n, where n is the size of the pattern used to generate the pyramid.

Each node in a pyramid has a pyramid address which is specified by a three-tuple (i,j,k). The last entry in the three-tuple, k, is the array number, or level of the node, in the pyramid. The first two entries, i and j, give the node's column and row numbers within array k. An hexagonal array has three major axes; two of these, x and y, are chosen in each array for enumerating rows and columns (see Figure 5).

The axis of each array will be rotated with respect to the next lower array in the pyramid. The angle of rotation, $\Delta\theta$, is determined by the pattern size, n (see Table 1), although this rotation may be either clockwise or counterclockwise. Let θ_k

be the rotation of the axis in array k relative to array 0. Then θ_k will be $k\Delta\theta$ if all rotations are counterclockwise and $-k\Delta\theta$ if they are all clockwise. However the direction of rotation may be alternated from level to level, in which case $\theta_k = 0$ for k even and $\pm\Delta\theta$ for k odd.

5. Tiles defined for nodes of the pyramid

Thus far we have considered the problem of tessellating the hexagonal grid with a pattern of nodes such that the centroids of the tessellation patterns become grid nodes of a new hexagonal grid. It is possible to replace each node with a tile such that the tiles for the node at each level of the pyramid tessellate the plane. To do this we simply let the tile for the nodes in array 0 be the hexagon which has a diameter equal to the node separation at this level. Then the tile for a node at level k, k>0, is defined as the union of the tiles of the level k-1 nodes in the pattern which is centered at the level k node. Such tiles are shown in Figure 6 for k = 0, 1, and 2 for the compact size-7 (hexagonal) pattern. (Successively higher level tiles are outlined with successively broader lines.)

Tiles defined in this way always tessellate the plane, as may be shown by induction. It is well known that hexagonal tiles tessellate the plane (level k = 0). Suppose the tiles at level k tessellate the plane. We then group together those tiles associated with the nodes of the pattern used to generate the pyramid. Since, by definition, this pattern tessellates the grid at this level, every tile is included in exactly one group, so these groups must also tessellate the plane. These groups are just the tiles for level k+1 nodes, so level k+1 tiles also tessellate the plane.

Note in Figure 6 that the tiles formed by the hexagonal pattern (n=7) are approximately, but not exactly, hexagonal except at level 0.

6. Tree Formation

Tree structures which are analogous to the "quadtree" may now be defined on pyramids generated by admissible patterns. Since these trees are developed as a code for binary images, we first associate a binary value with each level zero node; its tile is either "black" or "white" as is appropriate for the hexagonally sampled image. Next, we select a high level node which has a tile that completely encompasses the image or area of interest. This node becomes the root of the tree. Let N be the level of the root node. The root is given a value of "black" if its tile is entirely black, or "white" if its tile is entirely white. Otherwise its value is "gray". We associate a set of n branches, or pointers, with a gray node, one for each level N-1 node in the generator pattern centered at the root. Each of these nodes then becomes the root of a subtree of the tree. Subtrees are assigned values of "black", "white", or "gray" in the same way. If a subtree has value black or white then that subtree is a leaf and has no branches. Each gray subtree has n branches and subtrees of its own. The tree is complete when all of its branches end in leaves.

Each node in a tree has a unique "tree address". To obtain this address, we label the branch at each node by a letter 'a' 'b' 'c' ... according to the label given to that node in the generating pattern. The tree address for a node at level k is

a sequence $(b_N, b_{N-1}, \ldots, b_{k+1})$ of labels for the branches one must follow to move from the root node at level N to the destination node at level k.

7. Translation between pyramid and tree addresses

An unfortunate aspect of the tree and pyramid structures defined for hexagonal grids is that there is no simple relation between the tree address of a node and its pyramid address, at least for cases in which there is an axis rotation between levels (e.g. for n = 3 or 7). Therefore, to finish our examination of these data structures we shall describe a procedure for translating from a pyramid to tree address in the case of the "septtree", that is, when the generator is the hexagonal size-7 pattern.*

Suppose we wish to find the tree address $(b_N, b_{N-1},\ldots,b_{k+1})$ for a node with pyramid address (i_k,j_k,k), which is assumed to fall within the domain of the tree with root at level N. Our procedure will be to first find the pyramid address $(i_{k+1},j_{k+1},k+1)$ of the level k+1 node which is the centroid of the pattern covering node (i_k,j_k,k). The position of the node in the pattern is then b_{k+1}. The procedure is repeated to find progressively higher level branches. We need only specify how the centroid address and branch number are obtained between the k and k+1 levels.

Assume that the principal axes of levels k and k+1 are centered on a unique "origin" node at level k. If the axis rotation is clockwise, as in Figure 5, then we observe that k+1 level node (n,m,k+1) falls on top of k level node $(\hat{n},\hat{m},k)$ where

*The reader is referred to [7] for an elegant algebra defined within the tree address domain.

$$n = 2\hat{n} + \hat{m}$$
$$m = 3\hat{m} - \hat{n} \qquad \text{(Eq. 1a)}$$

(These relations may be checked with the aid of Figure 5.) If the rotation is counterclockwise then this relation is given by

$$n = 3\hat{n} - \hat{m}$$
$$m = \hat{n} + 2\hat{m} \qquad \text{(Eq. 1b)}$$

We are given (i_k, j_k, k) and wish to find the centroid node $(i_{k+1}, j_{k+1}, k+1)$. In general this node will not be directly above the level k node in the sense described above. However, equations (1a) or (1b) can be used to obtain the level k+1 node nearest to the given level k node, and this nearest node will be the desired centroid.

For clockwise rotations:

$$i_{k+1} = \text{Round}[(3i_k - j_k)/7]$$
$$j_{k+1} = \text{Round}[(i_k + 2j_k)/7]$$
$$r = [i_k + 2j_k]_{\text{MOD7}} \qquad \text{(Eq. 2a)}$$

And for counterclockwise rotations

$$i_{k+1} = \text{Round}[(2i_k + j_k)/7]$$
$$j_{k+1} = \text{Round}[(3j_k - i_k)/7]$$
$$r = [2i_k + j_k]_{\text{MOD7}} \qquad \text{(Eq. 2b)}$$

Here function "Round" simply rounds its argument to the nearest (larger or smaller) integer value. The MOD7 function returns the remainder after its argument is divided by 7.

The remainder, r, may be used to obtain the branch b_{k+1}. Let the nodes of the size 7 generator pattern be labeled 'a' 'b' ... 'g' in the order shown for one such pattern in Figure 5. The correspondence between r and b_{k+1} is given in Table 2.

Once b_{k+1} and $(i_{k+1},j_{k+1},k+1)$ have been obtained from equations (1) or (2) and Table 2, the process is repeated to find b_{k+2}, $(i_{k+2},j_{k+2},k+2)$, and all higher level branch labels and node addresses until the root is reached at level N, and the tree address of (i_k,j_k,k) is complete.

8. A Final Observation

Shmuel Peleg suggests that a good pattern for generating a tree or pyramid is one in which there is no node outside the pattern which is closer in geometric distance to its centroid than any node within the pattern. An outside node is allowed to be at the same distance as an inner node. This is true for the compact patterns of size n = 3, 4, and 7 in Figure 4. However, if we apply the same criterion to higher level tiles we see that the constraint is always violated if we move to a sufficiently high level. For example a violation occurs at level 3 for the n = 3 and 7 patterns and at level 2 for n = 4. A violation occurs for the standard quadtree as it is defined on a rectangular grid at level 3. Of these patterns the one which generates the largest tile without violations is the hexagonal n = 7 pattern. At level 2 this covers 49 level 0 nodes. By contrast the largest coverage of a standard quadtree is only 16 level 0 nodes. We conjecture that the hexagonal n = 7 pattern provides the largest compact tile, in the sense suggested by Peleg, of any admissible pattern defined on either hexagonal or rectangular grids.

References

[1] G. M. Hunter and K. Steiglitz. Operations on images using quadtrees. IEEE Trans. PAMI, 1, 1979, 145-153.

[2] A. Klinger and C. R. Dyer. Experiments in picture representation using regular decomposition. Computer Graphics and Image Processing, 5, 1976, 68-105.

[3] M. J. E. Golay. Hexagonal parallel pattern transformations. IEEE Trans. Comput., C-18, 1969, 733-740.

[4] A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. Pattern Recognition, 1, 1968, 33-62.

[5] S. L. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. Computer Graphics and Image Processing, 4, 1975, 104-119.

[6] E. M. Riseman and A. R. Hanson, Design of a semantically-directed vision processor. Tech. Rept. 74C-1, Dept. of Computer and Information Science, Univ. of Mass., Amherst, Mass., 1974.

[7] D. Lucas and L. Gibson, A system for hierarchical addressing in Euclidean space. Submitted to Proc. AMS.

Table 1: Admissible Pattern Sizes

node	distance	n	$\Delta\theta$
1	1	1	0
2	$\sqrt{3}$	3	30
3	2	4	0
4	$\sqrt{7}$	7	19
5	3	9	0
6	$\sqrt{12}$	12	30
7	$\sqrt{13}$	13	14
8	4	16	0

Table 2: Branch Label Determined from r

	r =	0	1	2	3	4	5	6
b	clockwise	a	b	d	c	f	g	e
	counterclockwise	a	d	b	c	f	e	g

Figure 1: Patterns which tessellate the plane but which violate constraints C2, C3 or C4.

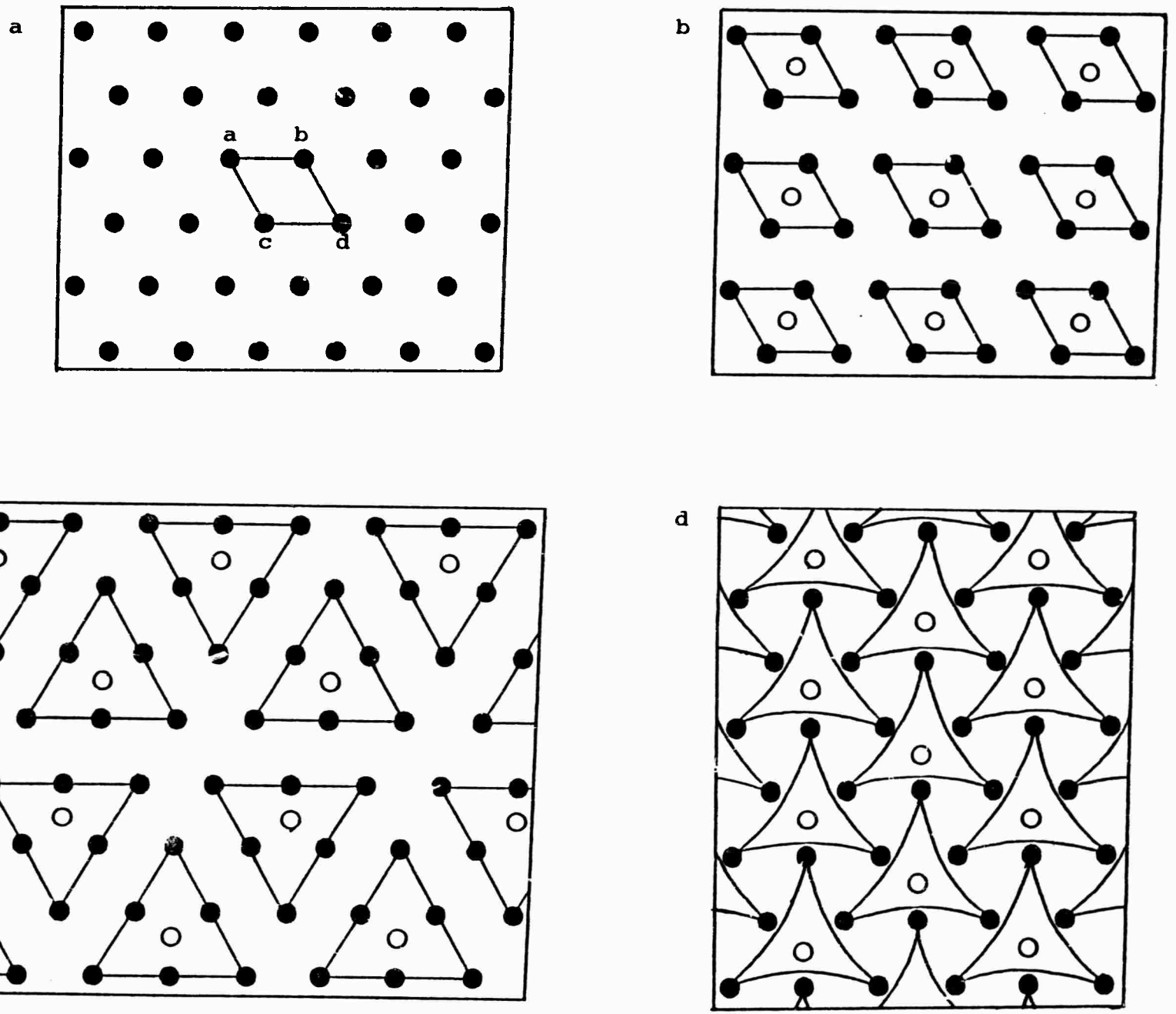


Figure 2: The unique spatial arrangement of a designated node in admissible patterns of size 3(b), 4(c), and 7(d).

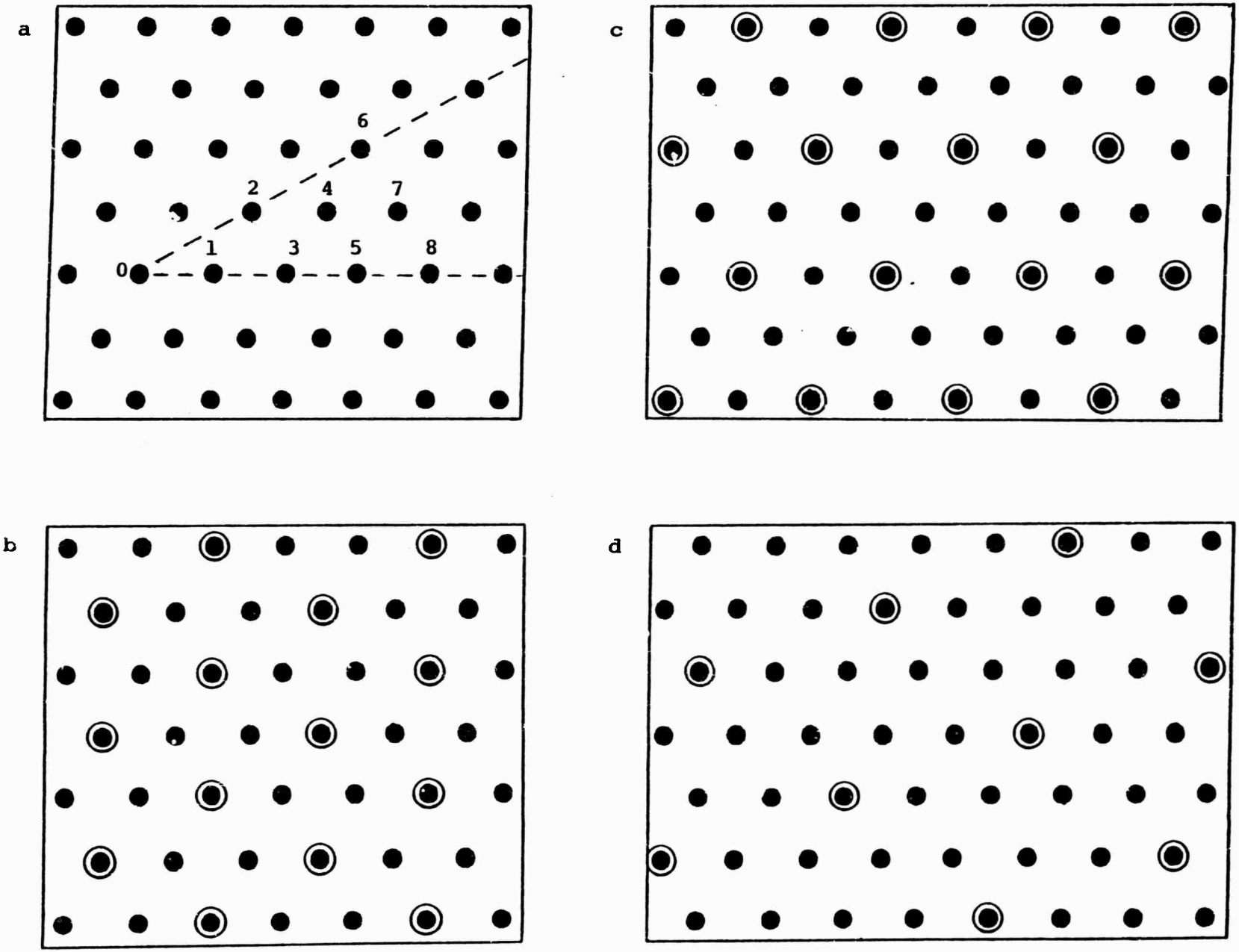
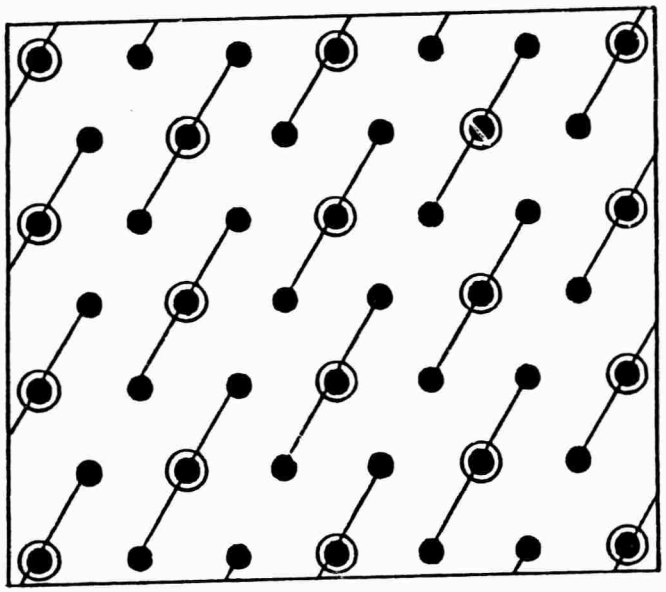
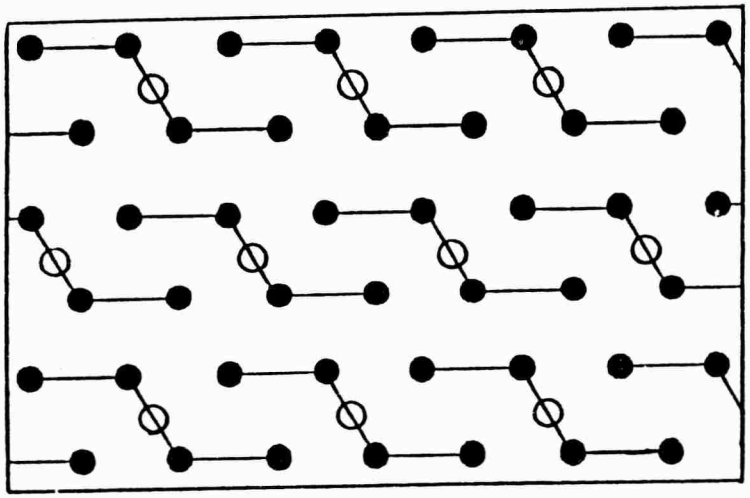


Figure 3: Examples of admissible patterns which are not optimally compact.

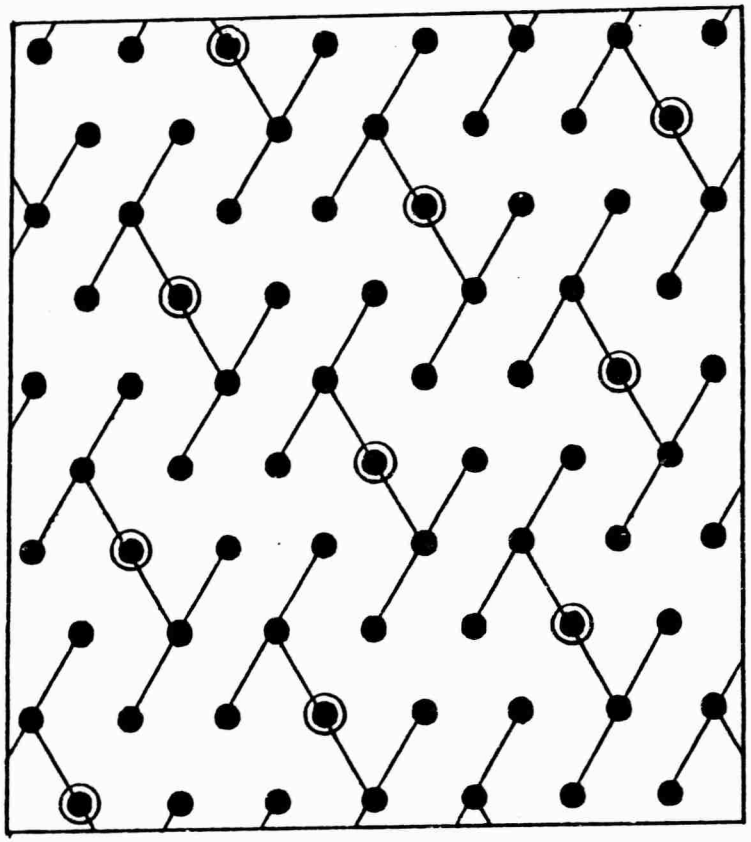
a

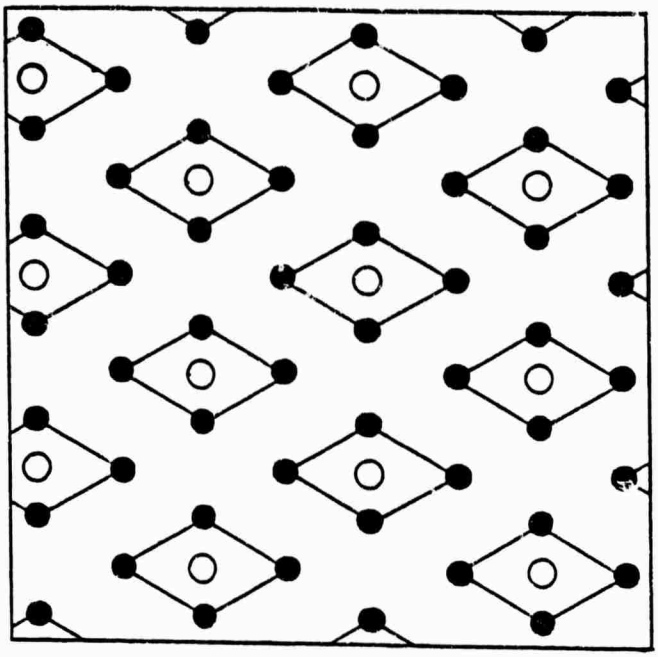


b

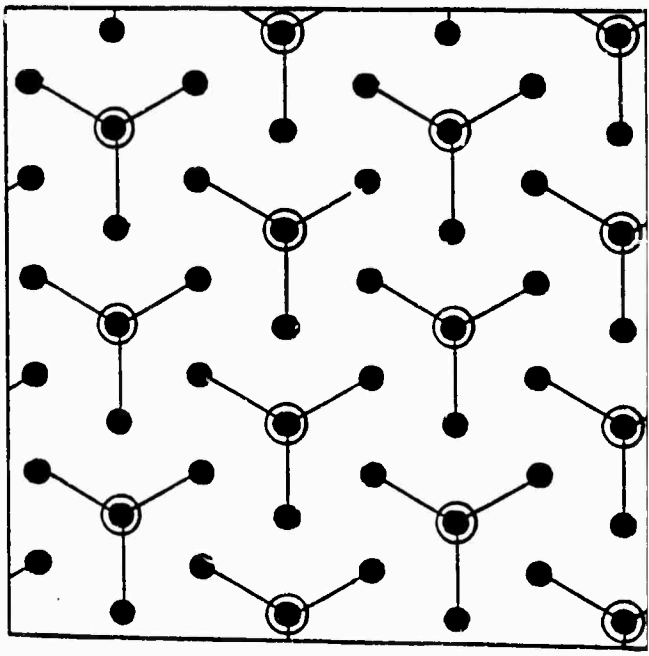


c

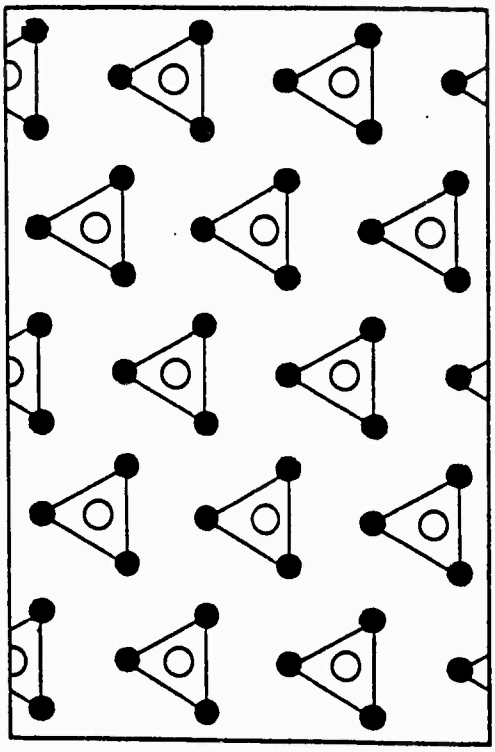




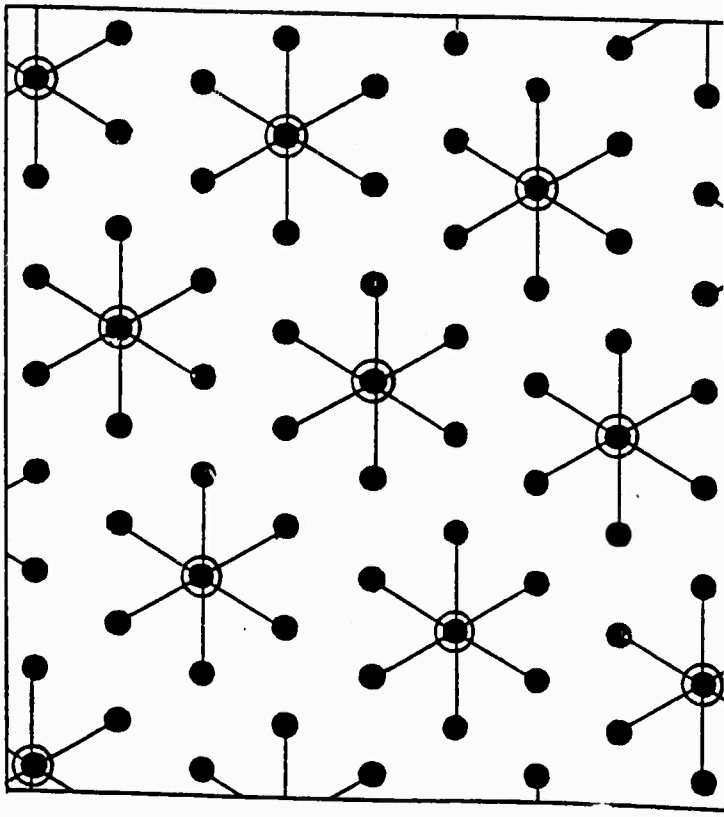
b



d



a



c

Figure 4: Compact admissible patterns of size 3(a), 4(b), and 7(c). Figure d shows an alternative size 4 patt rn which is "star" symmetric.

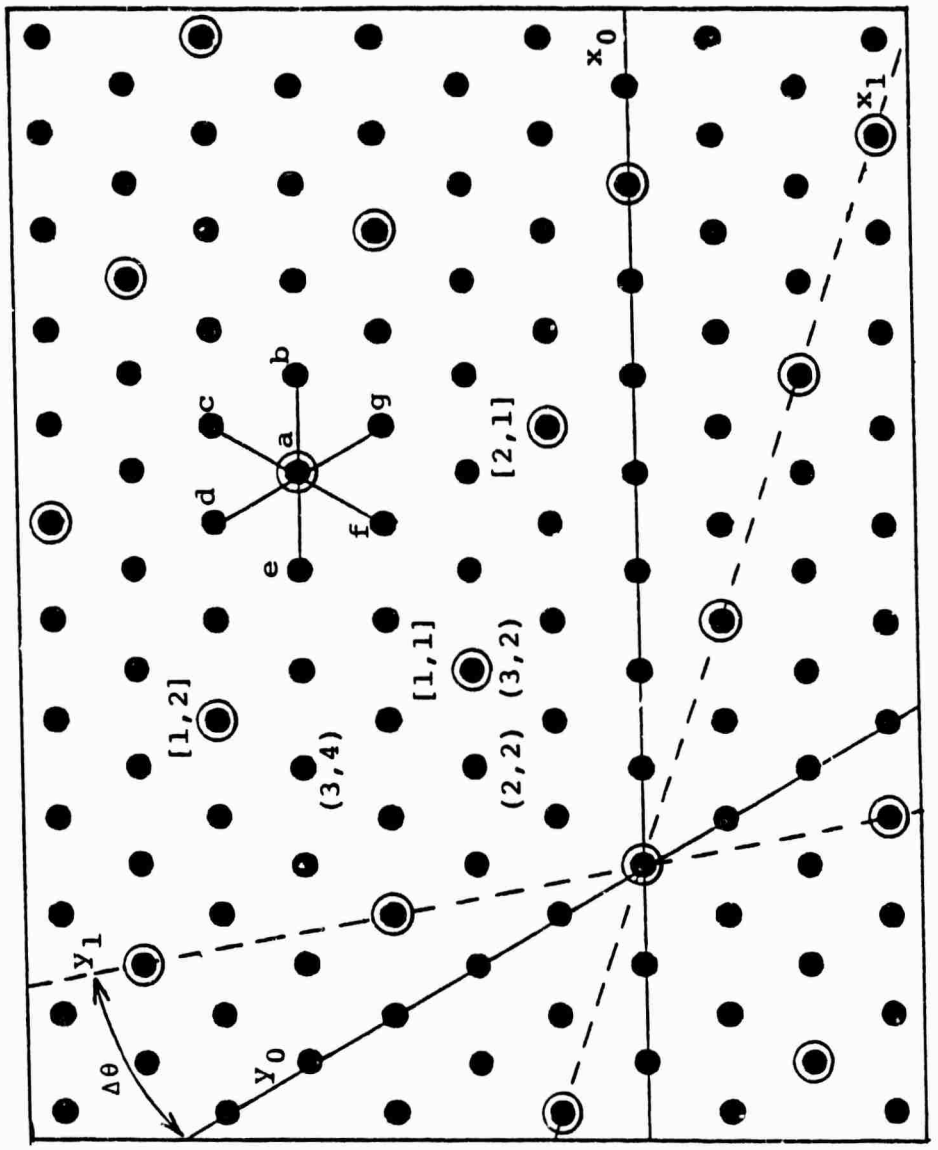


Figure 5: Coordinate system for specifying node locations in hexagonal grids. Open circles show the centroids of compact size 7 patterns used to tessellate the grid of black circles. Note that the axes of the grid of centroids is rotated by 19.1 degrees with respect to that of the original grid. Pairs of numbers in parentheses show the x,y location for a few closed circles. Pairs in brackets show locations of centroids.

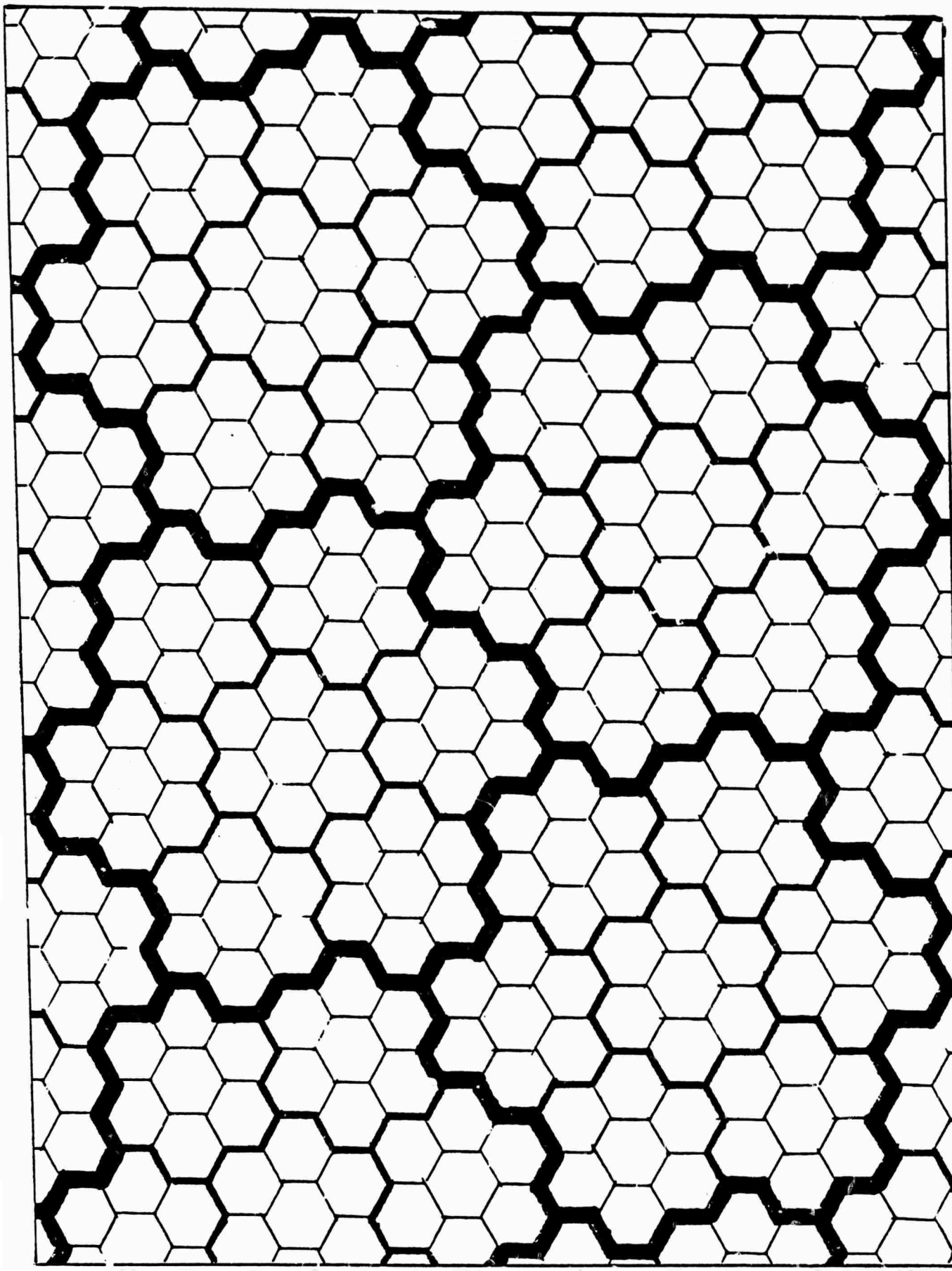


Figure 6. Hierarchy of tiles for the size 7 compact pattern. Tiles for level 0, 1 and 2 nodes are shown by progressively thicker outlines

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 099	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TREE AND PYRAMID STRUCTURES FOR CODING HEXAGONALLY SAMPLED BINARY IMAGES		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER TR-814
7. AUTHOR(s) Peter J. Burt		8. CONTRACT OR GRANT NUMBER(s) DAAG-53-76C-0138
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Night Vision Laboratory Ft. Belvoir, VA 22060		12. REPORT DATE October 1979
		13. NUMBER OF PAGES 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Hexagonal grids
Pyramids
Tree structures

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Tree and pyramid type data structures are described which may be used for storing and processing binary images that have been sampled on a hexagonal grid. These are analogous to the quadtrees and pyramids which have recently been developed for images sampled on a rectangular grid. Trees may be formed with 3, 4, 7, 9 ... branches per node. Of these the "septtree", formed with 7 branches per node, promises to be particularly interesting since each node "covers" a roughly hexagonal region of the image.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)